Øving 8, praksis: Traversering av grafer

TDT4120 – Algoritmer og datastrukturer 2018

Question 1: Konstruer grafen

I denne oppgaven skulle man opprette et `Node`-objekt for hver liste i `adjacencylist` og legge til naboene for hver av disse nodene.

Her blir det enklest å ***først*** opprette alle nodene og deretter legge til naboer. Man kan opprette alle nodene på flere måter.

Med en for-løkke:

```
nodelist = []
for i in 1:length(adjacencylist)
    push!(nodelist, Node(i))
end
```

Med *list comprehension*:

```
nodelist = [Node(i) for i in 1:length(adjacencylist)]
```

Eller med Julias dot-notasjon:

```
nodelist = Node.(1:length(adjacencylist))
```

Etter at man har opprettet alle nodene, kan man gå igjennom alle naboene for hver node og legge til riktig nabo i `neighbours`-listen. En fullstending implementasjon kan se ut som dette:

```
function makenodelist(adjacencylist)
    nodelist = Node.(1:length(adjacencylist))
    for node in nodelist
        for neighbourid in adjacencylist[node.id]
            push!(node.neighbours, nodelist[neighbourid])
        end
    end
    return nodelist
end
```

Question 2: Redd liv med bredde-først-søk

Her skulle man implementere BFS for å finne en ***forøkende sti*** i en bipartitt graf som igjen ville gitt en matching i en donor-resipient-par. Selv om denne oppgaven har store anvendelser, trengte man ikke å implementere noe særlig utover pseudokoden til BFS i pensumboken på side 595.

Her er en implementasjon i Julia:

```
function bfs!(nodes, start)
        for node in nodes
                node.color = "white"
                node.distance = typemax(Int)
        node.predecessor = nothing
        end
        start.color = "gray"
        start.distance = 0
        queue = Queue{Node}()
        enqueue!(queue, start)
        while !isempty(queue)
                current = dequeue!(queue)
                if isgoalnode(current)
                        return current
                end
                for neighbour in current.neighbours
                        if neighbour.color == "white"
                                neighbour.color = "gray"
                                neighbour.distance = current.distance + 1
                                neighbour.predecessor = current
                                enqueue!(queue, neighbour)
                        end
                end
                current.color = "black"
        end
end
```

Question 3: Redd liv med bredde-først-søk

I denne oppgaven skulle man finne en sti gitt at BFS allerede var kjørt på en graf. Man fikk kun sluttnoden som argument. Denne oppgaven minner veldig om oppgaven i praksisøving 2. Man kunne også hente inspirasjon fra PRINT-PATH i pensumboka på siden 601.

Denne oppgaven kan enten gjøres rekursivt (som PRINT-PATH) eller iterativt.

Her er en rekursiv løsning:

```
function makepathto(goalnode)
    if goalnode == nothing
        return Int[]
    end
    return push!(makepathto(goalnode.predecessor), goalnode.id)
end
```

Det går an å gjøre denne løsning enda kortere med en såkalt ***ternary operator*** (som du kan lese mer om her):

```
makepathto(n) = (n == nothing) ? Int[] : push!(makepathto(n.predecessor),
n.id)
```

Her er en iterativ løsning:

```
function makepathto(goalnode)
    path = Int[goalnode.id]
    nextnode = goalnode.predecessor
    while !(nextnode isa Nothing)
        push!(path, nextnode.id)
        nextnode = nextnode.predecessor
    end
    return reverse(path)
end
```